Promotion of Educational Effectiveness by Translation-based Programming Language Learning Using Java and Swift

Juhua LI
Waseda University, Tokyo, Kanto, Japan
khlee1290@gmail.com

Kazunori Sakamoto
National Institute of Informatics
exkazuu@nii.ac.jp

Hironori Washizaki
Waseda University, Tokyo, Kanto, Japan
washizaki@waseda.jp

Yoshiaki Fukazawa
Waseda University, Tokyo, Kanto, Japan
fukazawa@waseda.jp

Abstract

More and more programming tools have been created to help people to learn new programming languages. Although the number of tools to support beginning learners has increased, none directly compare different languages. This paper proposes a translation-based programming learning method that supports programming language learning for beginners of a new language who are familiar with a different language. Comparing the same code written in the two languages allows learners to discover commonalities and differences between the two languages, understand grammar rules, and successfully write programs in the new language. Our method is demonstrated using a web-based educational environment that translates Java into Swift. An experiment to evaluate the educational effectiveness confirms that using a programming language educational environment with translation support can aid beginning learners in learning a new programming language.

1. Introduction

With the soaring population of programming learners, the number of programming language learning tools has rapidly increased. Common programming learning tools include guidebooks, official documents, tutorial websites, and online academies.

If a learner is familiar with a general text-based programming language, he or she already has Computational Thinking [4], which "involves solving problems, designing systems, and understanding human behavior by drawing on fundamental computer science." Such learners already possess knowledge about structure, algorithm, and Computer Thinking, which are necessary to solve a problem or a task. Thus, when learning a new programming language, they tend to focus on the basic grammar of the new programming language.

For such learners, a comparison is an effective domain-general means of learning [11]. A comparison is intended to foster "structural alignment" of two objects, consequently allowing important common properties to be discovered. Since the process of aligning two representations can extract a common higher-order relational structure, a comparison can help extract abstract knowledge for a wide range of tasks. Because comparison processes can promote conceptual learning, it may also assist in learning a new programming language.

According to Henrik [11], if the same information of code written in two different representations (programming languages) is compared, then the relevant relations between the representations can be extracted, allowing crucial structural features to be understood. In other words, when learning a new language, learners can discover the commonalities and differences between two languages by comparing the same meaning of code written in two different languages to build knowledge of the grammar rules in the new language. Comparing a new language to a familiar language should increase learning productivity.

However, most current learning tools only include explanations and examples of one specific programming language. A solution to this problem is to employ some form of a translation approach. This paper proposes a translation-based programming language learning method, which provides a

URI: http://hdl.handle.net/10125/41166
ISBN: 978-0-9981331-0-2

translation between two languages to support mainstream learners learning a new language. As an implementation, an educational environment is designed as a web-based programming language learning application called Jasmin, which provides translation support between two languages.

In this research, two ways are used to verify that this environment enhances the understanding and coding skills of learners: reading comprehension and writing codes. Reading comprehension means the knowledge of a piece of code can be understood and appropriately applied, whereas writing code means that coding in a new programming language can be constructed to solve a given task. Reading and writing methods are commonly used at almost all universities to assess students' knowledge levels.

Therefore, the research questions in this research are:

RQ1: Compared to common programming learning tools, does language translation support in Jasmin promote the learners' educational effectiveness when learning a new programming language?

RQ2: Compared to common programming language tools, which part of translation support in Jasmin is more useful: initializing knowledge or starting to program in new languages?

The results show that an educational environment with translation support can aid beginning learners in learning new programming languages. Additionally, this environment more effectively promotes understanding the grammar rules of a new programming language than initializing coding skills.

As a demonstration, we translate Java and Swift in Jasmin, which are an Android application developing language and an iOS application developing language, respectively. We selected these languages for the following reasons. First, mobile applications have become very popular, and many people are interested in learning to program using these languages. "Of the 19 million software developers in the world, 8.7 million are now writing apps targeted for mobile devices, according Evans Data's recently released Developer Population and Demographics Study [6]. This represents a doubling of the mobile developer population since 2010 and an increase of 700,000 in the last year [6]." The population of mobile-minded programmers is definitely increasing.

Second, because the Android application market and the iOS application market have different platforms (Google Play and Apple Store), each application must have two versions to satisfy more users. In many companies, Android developers are asked to migrate to iOS developers (and vice versa) to more effectively manage human resources. Therefore, if Java can be easily translated into Swift, this would help Android developers migrate into iOS developers more smoothly.

The rest of this paper is organized as follows. Related works are listed in Section 2, while Section 3 describes the educational environment design and implementation with translation support. Section 4 details the experiment. Section 5 discusses the results, and Section 6 concludes the paper as well as provides future work.

2. Related work

Previous works have studied how to handle the translation approach to aid student learning new languages. For example, Matsuzawa [3] proposed an educational environment that offers improvements over current tools in order to help programming learners migrate from a visual language to a text language using Block to Java as an example. They developed a system that translates bidirectionally between Block (the block language used here) and Java [3]. The results indicate that the environment can act as scaffolding in the migration phase. This research successfully promoted learner's seamless migration from Block to Java.

Dann [2] also proposed an Alice to Java approach to encourage students' continuing to learn computing in computer science. They transferred an identical Alice program example directly into the Java program to mediate the transfer of concepts. Consequently, the transfer improved student achievement in learning Java [19].

Google has also emphasized programming language learning using a translation. Google PencilCode [23] provides a translation between two visual-based programming languages in a bidirectional way. Another tool Google Blockly [12] translates the block language to multiple languages (JavaScript, Python, PHP, Lua, Dart) in a unidirectional way to support beginning learners' migration from visual-based languages to text-based languages.

Moreover, the number of programming language translators between text languages has rapidly expanded. According to Garnelis [20], Transifex offers a rough translation between two languages to localize applications and access new markets [20].

Qiu [21] developed a programming language translator from C to C++, and Java to C++ to reuse existing code, improve the quality of the software, and enhance the productivity of software engineers.

These previous works indicate that the demand of translation support has yet to be satisfied. Additionally, the translation between two languages truly helps learners migrate smoothly from one language to another.

However, much of the current research for education is mainly focused on translation from visual-based languages to text-based languages. The main area of these efforts is introductory education for non-CS students. Research on translation between text-based languages focuses more on business usages rather than education. Thus, the educational effectiveness of the translation support between two text-based languages has yet to be verified.

The trend in programming language learning is learning text-based languages. Mainstream learners are computer science students, professional programmers, and developers. Hence, in this research, we propose a translation-based programming language learning method, which translates from a text language to another text language to support mainstream learners of new languages. As an implementation, an educational environment is designed as a web-based programming language learning application.

In this research, we confirm the educational effectiveness of translation support using two different methods: reading comprehension and writing codes. The results show that the translation support between two text-based programming languages not only helps increase the understanding of new grammar rules, but also helps learners start coding in a new language. The promotion in initializing coding skills is higher than the promotion in understanding grammar rules.

3. Educational environment design and implementation

The experiential learning [17] method is used as one of the concepts in Jasmin. Experiential learning [17] is the process of learning through experience; specifically, it is defined as "learning through reflection on doing." Knowledge and skills are added to a learner's experience by doing the learning activity and reviewing what was done [18].

Providing an experience and letting children enjoy what they are doing is more effective than presenting theories that have to be memorized and giving examples, as pointed out by Torin [22] from an interview about the Association of Waldorf Schools of North America. He identified that through experience children can generate ideas of some theories by themselves. For example, over time third-grade children who were allowed to go out construct buildings on the playground and go field trips were able to accurately compare different places and people around the world. Eventually they were able to expand this knowledge to think about how people live and to relate to one another.

Jasmin is designed to apply this kind of experiential learning to allow learners to decipher the commonalities and differences between two languages. Furthermore, Jasmin helps learners migrate smoothly from one language to another.

The learning process in Jasmin involves the following steps. First, learners are given reading comprehension questions and coding tasks to solve. Then a translation between Java and Swift is provided by Jasmin. Learners can use Jasmin, other references, and current tools to search for more detailed explanations of certain grammar. By solving these problems and tasks, learners can experience what a Swift program looks like and how to code in Swift. After solving these problems, learners develop basic knowledge of the Swift language.

To provide experience about understanding grammar rules and coding in Swift, Jasmin is divided into two parts. Figure 1 shows the first part, which can be further subdivided into three sections. The question section that addresses the reading comprehension questions about calculating the output of a piece of Swift code is displayed on the left. The translator section, which translates Java into Swift, is on the bottom. After a learner presses the "show me the answer" button, the correct answer is displayed in the answer section (Fig. 2).

The Java program, which is translated from the piece of Swift code in the question section, is displayed with the answer. To help learners compare the two languages and determine the commonalities and differences, the Java program is treated as an explanation of the Swift code. Presenting the Java code reduces the time cost. In the example shown in Fig. 2, we expect learners to become aware of what the range operator (…) in Swift means by simply looking at the Java code instead of searching for the "Range operator in Swift" among the references.

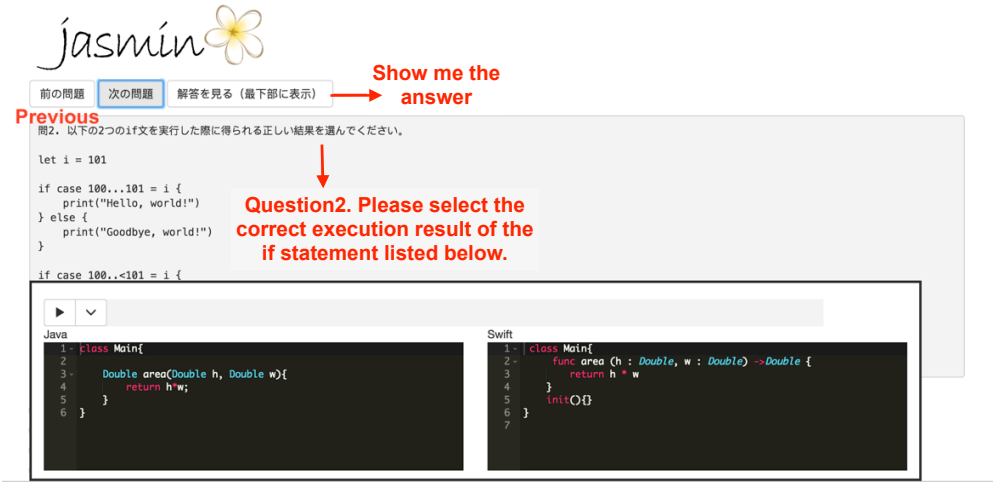

Figure 1. Reading comprehension part of Jasmin

The two editors are placed equally in the translator section. The Java programs are written in the left editor to translate for the same example as the Swift code. The target Swift code is generated immediately by pressing the start button on the left top, reducing the time that learners spend searching through documents or tools for certain Swift grammar rules.

Several features are required to implement this translation:

F1. Programs written in different languages are provided simultaneously.

F2. Statement sequences in both programs are identical.

F3. Programs in a new language are given immediately.

Obviously, the translation satisfies all these features. For example, the two programs are provided in two editors by the translator with the same statement sequence. The exact same examples are shown in the answers section.

Figure 3 shows the inner process of implementing the translator. Translation is divided into three main steps. The first step scans the inputted Java program to build up the Java abstract syntax tree (AST). Then a Java mapper is used to map Java AST to the Unified Code model. Finally, the Swift generator traverses and modifies the Unified Code model to adjust to the Swift grammar and print out the target Swift code on the screen.

The Java Mapper and JUNICOEN are employed to generate a Unified Code model from Java programs [5]. Java Mapper has two parts: a Java parser and a Java-to-Unified-code-model converter. The Java parser is automatically generated by ANTLR (Another Tool for Language Recognition) [2]. ANTLR as a parser generator not only recognizes languages but also interacts with a lexical analyzer (scanner), reports parsing errors, constructs abstract

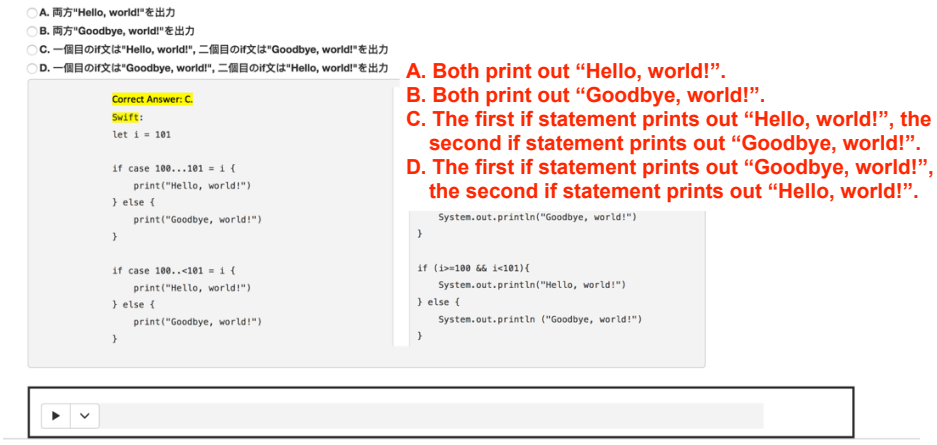

Figure 2. Translation provided in the answer

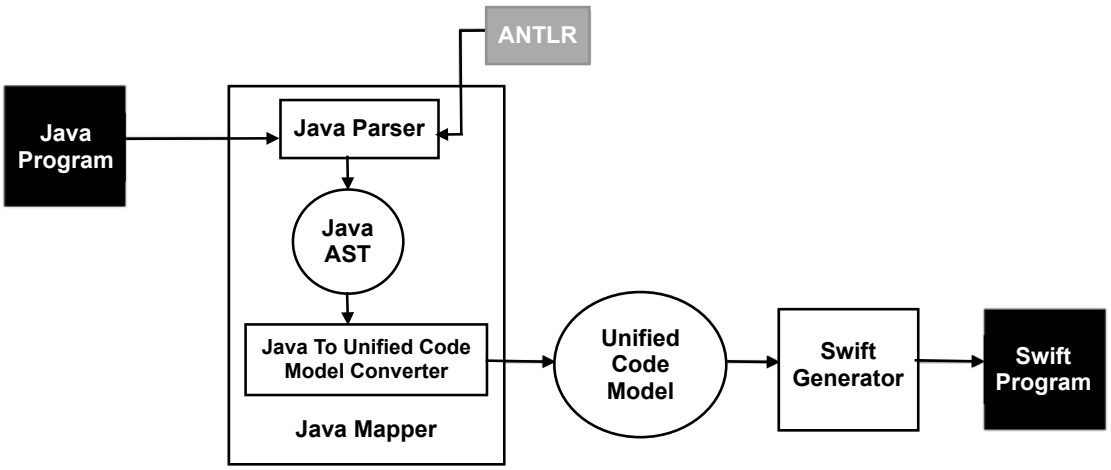

Figure 3. Inner process of the translator

syntax trees, and calls user actions [2]. Using a few simple grammar annotations, ANTLR parsers can automatically construct abstract syntax trees (AST), preventing users from explicitly calling the tree constructor routines [2]. Thus, the inputted Java programs can be scanned to build up Java abstract syntax tree (AST) by ANTLR [2].

Then JUNICOEN [5] builds the Unified Code model from Java AST. The Unified Code model is a language-independent AST, which has its own universal nodes (Uni-Nodes) to build up AST [5]. The Unified Code model is produced from Java AST by a Java-to-Unified-code converter. The Unified Code model is treated as a middle representation during translation. The Unified Code model can be extended to handle multiple programming languages in the future.

Then the Unified Code model is traversed by the Swift generator according to the Swift grammar to generate the target Swift program, which can be displayed on the screen. The purpose of the Swift generator is to traverse the Unified Code model and generate a translated, accurate, and runnable code to display on the screen. However, the grammar concepts of Java and Swift differ slightly in many places, including keywords, syntax, etc. For example, the constructor in Java is treated as a method that has the same name with the class name and has no return type. However, in Swift, the constructor has a specific name called "init". Sometimes switching the keyword is sufficient to produce the code, while other times an AST transformation is required to avoid mistakes when producing the Swift code. Deleting unnecessary keywords and elements as well as changing the necessary keywords and elements are included in the AST transformation to adjust to the Swift grammar.

Figure 4 shows the second part, which includes the question section, answer section, and translator section. The translator can be used to solve the tasks, while pressing the "show me the answer" button displays the solution.

Since the purpose of this research is to help beginning learners learn a new programming language more smoothly, only basic grammar of the new language, including class declaration, field declaration, function declaration and recall, constructor, if and switch statements, optional value and if-let statement, range operator, closure, dictionary declaration, type cast, array declaration, and access, is handled in this educational environment. These basic grammar rules become the starting point to learn a new language. After obtaining knowledge of the grammar rules, learners can understand and write more difficult Swift programs, including UI managements.

4. Experiment

4.1. Experimental design

An experiment was conducted to evaluate the educational effectiveness of our translation-based programming learning method in terms of reading comprehension and coding skills. Twelve subjects with some experience in programming were invited through Twitter to participate. The subjects consisted of students, programmers, and other people from different jobs. All were familiar with Java, but none

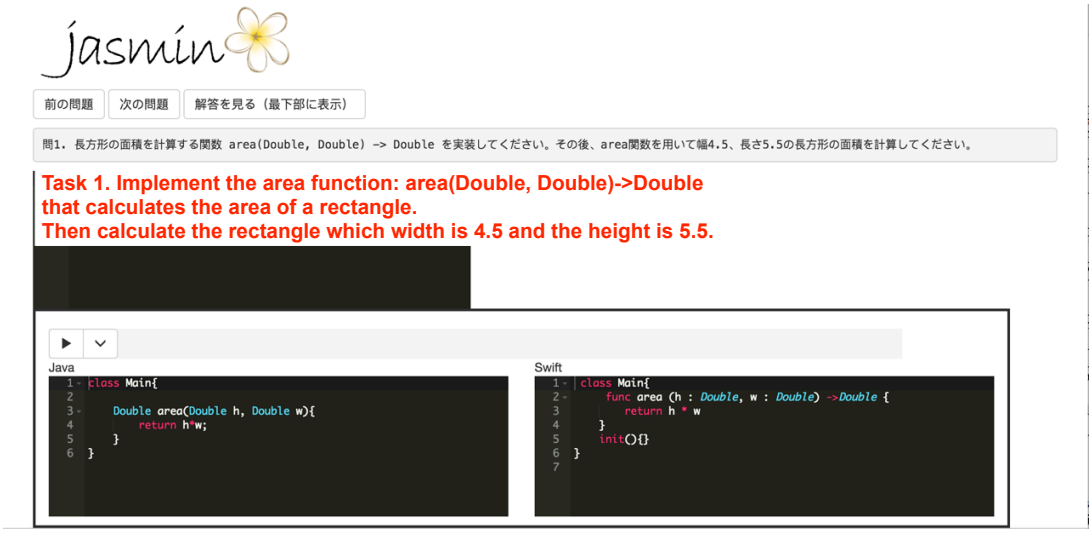

Figure 4. Coding task part of Jasmin

had experience with Swift. They were randomly divided into two groups: A and B. Group A did not have translational support, but group B did. Group A had seven subjects, whereas group B had five subjects.

All participants received the same initial reading comprehension questions and coding tasks. Then Jasmin with translation support was provided to one of the groups (group B). The other group (group A) received only the correct answers of the questions and tasks without translations between the two languages.

Every subject was allowed to self-study. Both groups were asked to solve the questions and tasks during the self-study section. Swift tutorial documents [15] from Apple were distributed to every subject as a standard reference. Additionally, all kinds of published programming learning tools and documents could be used during the self-study section to learn the basic grammar rules and the Swift language.

The time cost of the self-study section was recorded. Then to verify the learning performance, each subject completed a blind test, which included reading comprehension questions and coding tasks that were in the same form as the problems distributed before the self-study session. The questions and tasks were mainly from the “Test Your Swift” website [10] and “Programming Language Swift Definition Guide 2nd Edition” book [9]. None of the references (Xcode application, online compiler, programming language learning tools) were allowed during this test. The total score of both parts added up to ten.

Finally, the subjects were asked to answer a questionnaire, which was designed to analyze attitudes towards this experiment, the Swift language, and translation support. The group without translation support was allowed to access Jasmin after the test, allowing them to experience translation support in the learning process. They were also asked about their thoughts of this kind of support.

Table 1 illustrates the required language knowledge in the self-study section to solve the questions and tasks as well as the knowledge tested in the blind test.

4.2. Experimental results

Table 2 shows the measurements of the learning performance for the group without translation. Each subject had a drastically different score. Almost all subjects earned a better score in the reading comprehension section. Additionally, the total score in reading comprehension was higher than that for coding.

Table 3 shows the learning performance of the group with translation. 80% of the subjects in this group attained the highest score. Interestingly, the time cost differed significantly in this group. B2 earned the total highest score with only 32 minutes of time cost, while subject B4 used over 2 hours during the self-study section.

Table 1. Language knowledge covered in Jasmin and the experiment

	Language knowledge required to acquire in self-study section	Language knowledge tested in reading comprehension and coding tasks
Array access	✔	✔
Array declaration	✔	✔
Basic data types(Int, Double, Float, String, Character, Boolean)	✔	✔
Class declaration	✔	✔
Closure	✔	✔
Constructor	✔	
Dictionary declaration	✔	✔
Dictionary count property	✔	
Enum declaration		✔
Enum element access		✔
Field declaration	✔	✔
For loop	✔	✔
Function declaration	✔	✔
Function recall	✔	✔
If statement	✔	✔
If-case statement	✔	
If-let statement	✔	✔
Nested functions	✔	✔
Object creation	✔	✔
Optional value	✔	✔
Pattern matching operator	✔	
Range operator	✔	✔
Return statement	✔	✔
String library functions	✔	✔
Switch statement	✔	✔
Ternary operator	✔	
Type cast	✔	✔
Variable, constant declaration	✔	✔

Table 2. Measurements of the learning performance without translation support

	Subject A1	Subject A2	Subject A3	Subject A4	Subject A5	Subject A6	Subject A7
Study time (min)	102	34	79	57	139	19	43
Reading score	9	7	6	7	6	8	9
Programming score	8	3	8	3	6	6	4

Table 3. Measurements of the learning performance using translation support

	Subject B1	Subject B2	Subject B3	Subject B4	Subject B5
Study time (min)	34	32	112	200	39
Reading score	8	9	9	8	9
Programming score	5	7	7	8	7

Figure 5, 6, and 7 compare each part between the two groups. The left (right) side represents group A (B). Group A tends to have lower scores than group B in both reading comprehension and coding tasks. The minimum score, medium score, and the highest score for group A are lower than those of group B. In contrast, group A has a lower time cost than group B in the self-study section.

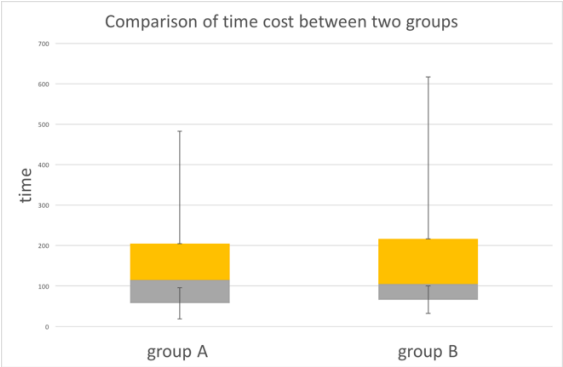

Figure 5. Comparison of the time costs between the two groups

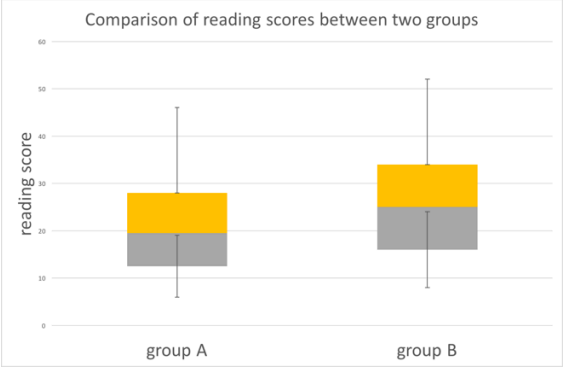

Figure 6. Comparison of the reading scores between the two groups

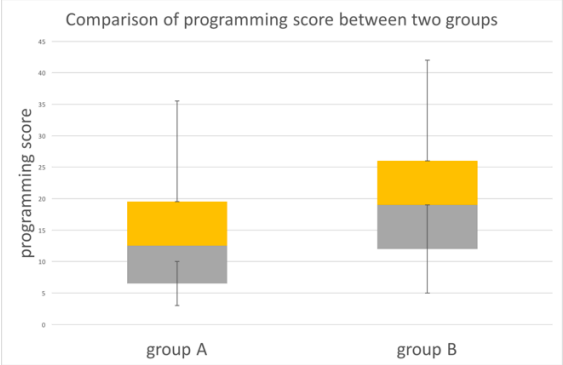

Figure 7. Comparison of the programming scores between the two groups

Table 4 compares the average of the learning performance between the two groups. Although group B spent about 16 more minutes in the self-study section than group A, the subjects in group B have significantly higher accuracy percentages in both the reading comprehension questions and the coding tasks. The differences in grammar understanding and coding skills between the two groups are 11.7% and 13.7%, respectively.

Table 4. Average of the study performance between the two groups.

	Group A	Group B
Average study time (min)	67.6	83.4
Accuracy percentage in reading	74.3%	86.0%
Accuracy percentage in programming	54.3%	68.0%

Table 5 shows the results of the questionnaire after the test. 91.7% of the subjects answered "yes" to the question about whether they compared the new language with a familiar language when learning or considering the solution in after seeing the tasks. About 92% of the subjects indicated that the translation from a new language to a familiar language is useful in both understanding new grammar and initializing coding skills. Additionally, 83.3% of the subjects had positive attitudes about the benefit of a translation from a familiar language to a new language when learning a new language. However, the subjects thought that a translation between two languages would help understand new grammar rules rather than coding in a new language.

Table 5. Results of the questionnaire after the test

		All Subjects
Translation existence		91.7%
From a new language to a familiar language	Help understand the grammar rules	41.7%
	Help coding in new languages	16.7%
	Help both	33.3%
From a familiar language to a new language	Help understand the grammar rules	58.3%
	Help coding in new languages	16.7%
	Help both	8.3%

5. Discussion

Table 2 and Table 3 suggest that coding in a new language is more difficult than understanding new grammar rules when learning a new programming language. The p value of the t-test is less than 0.01, which implies that learners often have more trouble with coding than understanding new grammar rules of a new language.

According to the Table 4, the accuracy percentage in both the reading comprehension questions and the coding tasks of group B are higher than group A. The

p value of the t-test in the reading comprehension part is less than 0.05, suggesting a statistical difference between the two groups. However, the p value in the coding part is larger than 0.05, indicating no statistical difference between the two groups.

Therefore, the answer to the first research question is positive. The translation support in Jasmin does promote the learners' educational effectiveness when learning a new programming language. Similarly, the answer to the second research question is affirmative. The translation support in Jasmin promotes initializing the knowledge part more than the coding part. The differences in programming experience and levels among subjects may lead to the conclusion that translation support is unable to help learners initializing coding skills of a new language.

Table 5 reveals that learners always compare a new language to familiar languages in order to discover the commonalities and differences as this can assist in a better understanding of the characteristics of a new language, extending the ability to think among diverse programming languages.

6. Conclusion and future work

To support learners to learn new programming languages, this paper proposes a translation-based programming language method. As a demonstration, the educational environment with a translation support called Jasmin was implemented and an experiment to evaluate the learning performance was conducted. Using a programming language educational environment with translation support can aid beginning learners in becoming proficient in a new programming language. Jasmin more effectively promotes understanding new grammar rules of a new programming language than initializing coding skills.

We received feedback about the experiment and translation support. One of the subjects thought Jasmin is good for the beginners to learn a new language because the translation support alleviates confusion and helps beginners program in a new language. Some of subjects felt that it is interesting to compare the specific grammar that Swift has but Java does not. They anticipate that this kind of translation support will promote the learning performance in the other programming languages. One of the comments asked about extending the current tool to cover more programming languages such as C++ or C#. Another comment pointed out that since learners can see the Swift code from the translator without bugs, they can concentrate more on learning a new language rather than fixing bugs, which improves motivation to keep learning Swift.

However, a few subjects admitted that they became confused when they were suddenly given questions without any grammar explanation. They confessed that some literal explanations about Swift grammar would be appreciated. Because Jasmin only provides translation support, learners have to search for more detailed explanations for certain grammar rules, reducing the motivation to learn.

There are several directions for future work. Based on the feedback, we plan to combine tutorial documents with Jasmin to fully support learning a new language. Jasmin could also be extended to cover more programming languages. Additionally, the relationship between language similarity and educational effectiveness could be considered as a future work. Herein Java and Swift, which are similar languages, are used to measure the educational effectiveness among students, but the learning effectiveness of the translator may drastically decrease if the two languages are completely different.

7. References

[1] Y.Matsuzawa, K.Sakamoto, T.Ohata, K.Kakehi , "Programming Language Translation System for Programming Education(In Japanese)", IPSJSIG-CE, August 2015, pp. 223-230.

[2] T.J.PARR, R.W.QUONG, "ANTLR: A Predicated-LL(k) Parser Generator", SOFTWARE-PRACTICE AND EXPERIENCE, VOL.25(7), JULY 1995, 789-810.

[3] Y.Matsuzawa, T.Ohata, M.Sugiura, S.Sakai, "Language Migration in non-CS Introductory Programming through Mutual Language Translation Environment", SIGCSE2015'Proceedings of the 46th ACM Technical Symposium on Computer Science Education, February 2015, Pages 185-190.

[4] Wing, J., Computational Thinking, Communications of the ACM, Vol. 49, No.3, 2006, pp. 33-35.

[5] K.Sakamoto, A.Ohashi, D.Ota, H.Washizaki, Y.Fukazawa, "UNICOEN : Framework that analyzes and transforms source code supporting multiple programming languages.(In Japanese)", IPSJ, Vol.54, No.2, (2013), pp. 945-960.

[6] Marthe Rana, "Mobile Developer Population Reaches 8.7M Worldwide, New Evans Data Developer Population and Demographics Study", available from <http://www.evansdata.com/press/viewRelease.php?pressID=210> (2016.05.30).

[7] Yang, T.-C., Hwang, G.-J., Yang, S. J. H., & Hwang, G.-H. "A Two-Tier Test-based Approach to Improving Students' Computer-Programming Skills in a Web-Based

Learning Environment". Educational Technology & Society, 18 (1), (2015), 198–210.

[8] Batia LAUFER, Haifa, Israel, "Electronic dictionaries and incidental vocabulary acquisition: does technology make a difference?", ELECTRONIC DICTIONARIES IN SECOND LANGUAGE COMPREHENSION.

[9] Takeshi Ogihara, "Programming Language Swift Definition Guide 2nd Edition", 2015/12/25.

[10] Test Your Swift, available from <https://www.hackingwithswift.com/test/> (2016.05.30) .

[11] Henrik Saalbach, Lennart Schalk, Michael Schneider, Elsbeth Stern, "Learning through Comparison", available from <http://www.ifvll.ethz.ch/research/Comparison> (2016.05.30).

[12] Google Blockly abaliable from <https://developers.google.com/blockly/ > (2016.05.30).

[13] Swift quiz, available from < http://www.ios-blog.co.uk/quiz/> (2016.09.11)

[14] Humaira, R.; Sakamoto, K.; Ohashi, A.; Washizaki, H. & Fukazawa, Y., "Towards a Unified Source Code Measurement Framework Supporting Multiple Programming Languages", Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute Graduate School, 2012, 480-485.

[15] The Swift Programming Language (Swift 2.2) from iOS Developer Library available from <http://www.studiogalago.com/the-swift-programming-language/ >(2016.05.30).

[16] The Java™ Tutorials from ORACLE Java Documentation available from <https://docs.oracle.com/javase/tutorial/java/index.html> (2016.05.30).

[17] Felicia, Patrick (2011). Handbook of Research on Improving Learning and Motivation. p. 1003. ISBN 1609604962.

[18] Experiential learning, Dick, Bob (2002) The design of experiential learning activities. Unpublished paper (mimeo).

[19]W. Dann, D. Cosgrove, D. Slater, D. Culyba, and S. Cooper. Mediated transfer, "Alice 3 to java". In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12, pages 141–146, 2012.

[20]Antonis Garnelis, "Translation between programming languages" [Online Document], cited August 2, 2016. Available HTTP: https://www.transifex.com/blog/2012/translating-between-programming-langs/ (2016.09.01).

[21] Lili Qiu, Programming Language Translation, Department of Computer Science Cornell University Ithaca, NY 14853

[22] Waldorf School of New Orleans, Experiential Learning and Waldorf Education, available from < https://www.youtube.com/watch?v=mSeyZFaGKIY> (2016.09.01)

[23] Google pencil, available from <https://pencilcode.net/ > (2016.09.01)